

DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL

IIIIII	NN	NN	DDDDDDDD	IIIIII	RRRRRRRR	EEEEEEEEEE	CCCCCCCC	TTTTTTTTTT
IIIIII	NN	NN	DDDDDDDD	IIIIII	RRRRRRRR	EEEEEEEEEE	CCCCCCCC	TTTTTTTTTT
II	NN	NN	DD	II	RR	EE	CC	TT
II	NN	NN	DD	II	RR	EE	CC	TT
II	NNNN	NN	DD	II	RR	EE	CC	TT
II	NNNN	NN	DD	II	RR	EE	CC	TT
II	NN	NN	DD	II	RRRRRRRR	EEEEEEEE	CC	TT
II	NN	NN	DD	II	RRRRRRRR	EEEEEEEE	CC	TT
II	NN	NN	DD	II	RR	EE	CC	TT
II	NN	NNNN	DD	II	RR	EE	CC	TT
II	NN	NN	DD	II	RR	EE	CC	TT
II	NN	NN	DD	II	RR	EE	CC	TT
II	NN	NN	DD	II	RR	EE	CC	TT
II	NN	NN	DD	II	RR	EE	CC	TT
IIIIII	NN	NN	DDDDDDDD	IIIIII	RR	EE	CC	TT
IIIIII	NN	NN	DDDDDDDD	IIIIII	RR	EE	CCCCCCCC	TT

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SSSSSS
LL	II	SSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LLLLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLLLL	IIIIII	SSSSSSSS

INDIRECT
Table of contents

- INDIRECT FILE MANIPULATION ROUTINES^{K 14}

15-SEP-1984 23:55:59 VAX/VMS Macro V04-00

Page 0

(2)	147	STACK INDIRECT FILE
(3)	272	DEFINE SYMBOLS P1-P8
(4)	308	PUSH PROCEDURE ONTO INDIRECT STACK
(5)	512	UNSTACK INDIRECT FILE SPECIFICATION
(6)	578	UNSTACK NEXT INDIRECT FILE
(7)	769	SAVE VERIFICATION STATE
(8)	805	RESTORE VERIFICATION STATE

```
0000 1 .TITLE INDIRECT - INDIRECT FILE MANIPULATION ROUTINES
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24
0000 25 *****
0000 26
0000 27
0000 28 D. N. CUTLER 2-MAY-77
0000 29
0000 30 INDIRECT FILE MANIPULATION ROUTINES
0000 31
0000 32 MODIFIED BY:
0000 33
0000 34 V03-016 HWS0100 Harold Schultz 06-AUG-1984
0000 35 Open any new indirect frame with Carriagecontrol
0000 36 attributes.
0000 37
0000 38 V03-015 HWS0081 Harold Schultz 15-Jul-1984
0000 39 When closing the current indirect frame and unstacking
0000 40 the previous frame, set NAM block to not use the RSA and
0000 41 ESA fields left by the indirect frame just closed. Using
0000 42 these values cause the free dynamic memory list to become
0000 43 corrupted. Add support for execute-only command procedures.
0000 44
0000 45 V03-014 HWS0080 Harold Schultz 12-Jul-1984
0000 46 When allocating room in symbol table for resultant
0000 47 name string, don't use constant of 256; use the value
0000 48 of NAM$C_MAXRSS+1 rounded up to a long word boundary
0000 49 instead. Remove the ASSUME of NAM$C_MAXRSS. Bypass
0000 50 deallocation of unused buffer if none to deallocate.
0000 51
0000 52 V03-013 HWS0066 Harold Schultz 21-May-1984
0000 53 Correct the error handling when a SYMOVF error is encountered
0000 54 while setting up the new indirect level. Reenable password
0000 55 masking after opening an indirect input file.
0000 56
0000 57 V03-012 HWS0015 Harold Schultz 21-Feb-1984
```



```
0000 58 : Check status after $FIND.
0000 59 : Initialize file spec. size fields in NAM block before reusing.
0000 60 : Deassign SYSS$INPUT prior to reopening a file at a prior indirect
0000 61 : level.
0000 62 :
0000 63 : V03-011 PCG0013 Peter George 12-Jan-1984
0000 64 : Fix broken branch.
0000 65 :
0000 66 : V03-010 PCG0012 Peter George 17-Aug-1983
0000 67 : Correctly clear RMS FS$SEARCH context.
0000 68 : Manage concealed logical name attribute using the new services.
0000 69 :
0000 70 : V03-009 PCG0011 Peter George 27-May-1983
0000 71 : Fix bug in unstacking when restored command procedure
0000 72 : is already positioned to EOF.
0000 73 :
0000 74 : V03-009 PCG0011 Peter George 27-May-1983
0000 75 : Fix bug in file name saving logic.
0000 76 : Fix bugs in SYSS$OUTPUT processing.
0000 77 :
0000 78 : V03-008 KRM0099 Karl Malik 29-Apr-1983
0000 79 : Disable password masking for network.
0000 80 :
0000 81 : V03-007 PCG0010 Peter George 10-Apr-1983
0000 82 : Finish making remote open work.
0000 83 :
0000 84 : V03-006 PCG0009 Peter George 22-Feb-1983
0000 85 : Add DCL$DEFINE_P1_TO_P8.
0000 86 : Clear FAB$M_SQD bit.
0000 87 : Clear FAB$V_NAM and FAB$W_IFI when performing
0000 88 : remote reopen.
0000 89 :
0000 90 : V03-005 PCG0008 Peter George 28-Jan-1983
0000 91 : Remove reference to ONEXIT bit.
0000 92 :
0000 93 : V03-004 PCG0007 Peter George 13-Jan-1983
0000 94 : Call SYSS$OUTPUT routines.
0000 95 : Save name of command procedure.
0000 96 : Use saved file name spec to reopen command procedures
0000 97 : on remote nodes.
0000 98 :
0000 99 : V03-003 PCG0006 Peter George 30-Dec-1982
0000 100 : Clear PRC_V_ONEXIT when unstacking.
0000 101 :
0000 102 : V03-002 PCG0005 Peter George 28-Oct-1982
0000 103 : Fix CLRBIT typo.
0000 104 :
0000 105 : V03-001 PCG0004 Peter George 15-Jul-1982
0000 106 : Allow execute-only command procedures.
0000 107 : ---
0000 108 :
0000 109 :
0000 110 : MACRO LIBRARY CALLS
0000 111 :
0000 112 :
0000 113 : PRCDEF ;DEFINE PROCESS WORK AREA
0000 114 : WRKDEF ;DEFINE COMMAND WORK AREA
```

INDIRECT
V04-000

N 14

- INDIRECT FILE MANIPULATION ROUTINES

15-SEP-1984 23:55:59 VAX/VMS Macro V04-00
4-SEP-1984 23:41:10 [DCL.SRC]INDIRECT.MAR;1

Page 3
(1)

```
0000 115 PTRDEF ;DEFINE RESULT PARSE DESCRIPTOR FORMAT
0000 116 IDFDEF ;DEFINE INDIRECT FRAME OFFSETS
0000 117 PRDDEF ;PROCESS RMS DATA
0000 118 SYMDEF ;DEFINE TYPES OF SYMBOLS
0000 119 $CLMSGDEF ;DEFINE ERROR/STATUS VALUES
0000 120 $DEVDEF ;DEFINE DEVICE CHARACTERISTIC BITS
0000 121 $FABDEF ;DEFINE FAB OFFSETS
0000 122 $RABDEF ;DEFINE RAB OFFSETS
0000 123 $LOGDEF ;DEFINE LOG OFFSETS
0000 124 $NAMDEF ;DEFINE NAM OFFSETS
0000 125 $PSLDEF ;DEFINE PROCESSOR STATUS FIELDS
0000 126
0000 127 :
0000 128 : LOCAL SYMBOLS
0000 129 :
0000 130
00000008 0000 131 SYMBOLS=8 ;MAXIMUM NUMBER OF INDIRECT FILE SYMBOLS
0000 132
0000 133 :
0000 134 : LOCAL DATA
0000 135 :
0000 136
00000000 0000 137 .PSECT DCL$ZCODE,BYTE,RD,NOWRT
0000 138 INPFILE: ; INPUT FILE DEFAULT NAME STRING
0000 139 .ASCII /.COM/ ;
54 55 50 54 55 4F 0000 140 OUTQUAL: .ASCII /OUTPUT/ ; REST OF NAME AND THE QUALIFIER
000A 141 SYS_INPUT NAME: ;
54 55 50 4E 49 24 53 59 53 00' 000A 142 .ASCII /SYSS$INPUT/ ; LOGICAL NAME FOR SYSS$INPUT
09 000A
0014 143 LNMS$PROCESS:
53 53 45 43 4F 52 50 24 4D 4E 4C 00' 0014 144 .ASCII /LNMS$PROCESS/ ; PROCESS LOGICAL NAME TABLE
0B 0014
3A 30 41 4C 4E 5F 00' 0020 145 NLAO: .ASCII /_NLAO:/ ; NULL DEVICE
06 0020
```



```
0027 147 .SBTTL STACK INDIRECT FILE
0027 148 :+
0027 149 : DCL$STACKIND - STACK INDIRECT FILE
0027 150 :
0027 151 : THIS ROUTINE IS CALLED TO STACK THE CURRENT INDIRECT FILE LEVEL AND TO PARSE
0027 152 : AND OPEN THE NEXT INDIRECT FILE.
0027 153 :
0027 154 : INPUTS:
0027 155 :
0027 156 : IT IS ASSUMED THAT THE INDIRECT FILE PROCESSING FLAG IS SET.
0027 157 :
0027 158 : OUTPUTS:
0027 159 :
0027 160 : THE CURRENT INDIRECT FILE SPECIFICATION IS SAVED ON THE INDIRECT FILE
0027 161 : STACK AND THE NEXT INDIRECT FILE IS PROCESSED.
0027 162 :
0027 163 : RO LOW BIT CLEAR INDICATES INDIRECT FILE PROCESSING FAILURE.
0027 164 :
0027 165 : RO = DCL$_ATLAST - INDIRECT FILE SPECIFICATION NOT LAST ITEM ON
0027 166 : COMMAND LINE.
0027 167 : RO = DCL$_DEFOVF - ATTEMPT TO DEFINE MORE THAN EIGHT PARAMETERS.
0027 168 : RO = DCL$_STKOVF - INDIRECT FILE INTERNAL STACK OVERFLOW.
0027 169 :
0027 170 : RO LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
0027 171 :
0027 172 : RO = DCL$_NORMAL - NORMAL COMPLETION.
0027 173 : -
0027 174 :
0027 175 DCL$STACKIND::
0027 176 BSBW SETIND
5E CO AE 9E 002A 177 MOVAB -(<SYMBOLS*8>(SP),SP
0027 178 CLRL -(SP)
F4BE CA D7 0030 179 DECL WRK L CHARPTR(R10)
0027 180 10$: BSBW DCL$MARK
53 03 9A 0037 181 MOVZBL #PTR K PARAMETR,R3
0027 182 BSBW DCL$PROCFILE
0027 183 BLBC R0,15$
0027 184 BSBW DCL$SETCHAR
0027 185 CMPB #^A/\,R0
50 2F 91 0043 186 BNEQ 20$
0027 187 BSBW DCL$MOVTOKN
004 51 D1 004B 188 CMPL R1,#4
0027 189 BLSS 13$
51 04 D0 0050 190 MOVL #4,R1
0027 191 13$: PUSHL R0
AA AF 62 51 29 0055 192 CMPC R1,(R2),OUTQUAL
0027 193 BEQL 14$
0027 194 POPL R0
0027 195 STATUS IVQUAL
0027 196 BRB 15$
0027 197 14$: POPL R0
50 3D 91 006B 198 CMPB #^A/=/,R0
0027 199 BEQL 10$
50 3A 91 0070 200 CMPB #^A/:/,R0
0027 201 BEQL 10$
0027 202 STATUS IVVALU
007D 31 007C 203 15$: BRW 80$
0027 204 :
```

```
:STACK INDIRECT FILE
:SET INDIRECT PROCESSING UP
:ALLOCATE SPACE FOR SYMBOL DESCRIPTOR
:CLEAR COUNT OF GENERATED SYMBOLS
:BACK UP TO AT SIGN
:MARK CURRENT PARSE POSITION
:SET TOKEN CONTEXT FOR FILESPEC
:PROCESS FILE SPECIFICATION
:IF LBC PARSE FAILURE
:PEEK AT NEXT CHARACTER IN INPUT BUF
:SLASH?
:IF NEQ NO
:MOVE TERMINATOR AND GET NEXT TOKEN
:MORE THAN MAX MATCH NAME
:BR IF NO
:ONLY CHECK FOR 4 CHARS
:SAVE TERMINATION CHARACTER
:CHECK FOR VALID QUAL
:BR IF OK
:RESTORE TERMINATION CHARACTER
:SET ILLEGAL QUALIFIER CODE
:
:RESTORE TERMINATION CHARACTER
:EQUAL SIGN TERMINATOR?
:IF EQL YES
:COLON TERMINATOR?
:IF EQL YES
:SET INVALID VALUE SYNTAX
:
```

```
007F 204
007F 205
007F 206 : FILE SPECIFICATIONS PARSED - PARSE SYMBOL DEFINITIONS
007F 207
007F 208 : IF THE FILESPEC WAS FOLLOWED BY A SPACE, THAT SPACE MAY HAVE BEEN THROWN
007F 209 : AWAY IF THE FIRST CHARACTER IN P1 MAKES IT INSIGNIFICANT.
007F 210
58 04 AE 9E 007F 211 20$: MOVAB 4(SP),R8 :GET ADDRESS OF SYMBOL DESCRIPTOR ST
    FF7A' 30 0083 212 :BSBW DCL$SETNBLK :IGNORE BLANKS AFTER FILESPEC
    FF77' 30 0086 213 30$: BSBW DCL$MARK :MARK POSITION OF FIRST NON-BLANK
    FF74' 30 0089 214 40$: BSBW DCL$MOVCHAR :COPY A CHARACTER FROM INPUT BUUFER
    04 E0 008C 215 :BBS #WRK_V_QUOTE,- :LOOP IF IN A QUOTED STRING
    F8 F0 AA 008E 216 :WRK_Q_FLAGS(R10),40$
    05 13 0091 217 :BEQL 45$ :BR IF END OF LINE
    50 20 91 0093 218 :CMPB #'A' ',RO :IS THIS A TERMINATOR
    F1 12 0096 219 :BNEQ 40$ :BR IF NO - KEEP LOOKING FOR TERMINA
    FF65' 30 0098 220 45$: BSBW DCL$MARKEDTOKEN :GET DESCRIPTOR OF PARAMETER
    51 D7 009B 221 :DECL R1 :REMOVE COUNT FOR TERMINATOR
    18 13 009D 222 :BEQL 60$ :IF NULL STRING - NO MORE SYMBOLS
    62 22 91 009F 223 :CMPB #'A/'',(R2) :SYMBOL START WITH A QUOTE
    03 12 00A2 224 :BNEQ 50$ :IF NO - LEAVE THE SYMBOL ALONE
    FF59' 30 00A4 225 :BSBW DCL$COMPRESS :ELSE REMOVE THE QUOTE PAIRS
    88 51 7D 00A7 226 50$: MOVQ R1,(R8)+ :STORE SYMBOL DESCRIPTOR
    DB 6E 08 F3 00AA 227 :AOBLEQ #SYMBOLS,(SP),30$ :ANY MORE SYMBOL DEFINITIONS ALLOWED
    00AE 228 :STATUS DEFOVF :SET SYMBOL DEFINITION OVERFLOW
    45 11 00B5 229 :BRB 80$
    00B7 230
    00B7 231
    00B7 232 : RUN DOWN ANY IMAGE CURRENTLY RUNNING
    00B7 233
    50 BA AA DD 00B7 234 60$: PUSHL WRK_L_RSLNXT(R10) :SAVE POINTER INTO WRK AREA
    FF43' 30 00BA 235 :BSBW DCL$RNDOWN :RUN DOWN IMAGE AND INDIRECT LEVELS
    BA AA 8E C3 00BD 236 :SUBL3 (SP)+,WRK_L_RSLNXT(R10),R0 :CALCULATE LENGTH OF STACK SHIFT
    68 AE 50 C0 00C2 237 :ADDL R0,<<SYMBOLS*8>+4+<9*4>>(SP) :RELOCATE SAVED WRK_L_RSLNXT
    6C AE 50 C0 00C6 238 :ADDL R0,<<SYMBOLS*8>+4+<10*4>>(SP) :RELOCATE SAVED WRK_L_RSLND
    70 AE 50 C0 00CA 239 :ADDL R0,<<SYMBOLS*8>+4+<11*4>>(SP) :RELOCATE SAVED WRK_L_EXPANDPTR
    74 AE 50 C0 00CE 240 :ADDL R0,<<SYMBOLS*8>+4+<12*4>>(SP) :RELOCATE SAVED WRK_L_MARKPTR
    00D2 241
    00D2 242 :
    00D2 243 : STACK COMMAND PROCEDURE
    00D2 244
    68 AE D0 00D2 245 :MOVL <<SYMBOLS*8>+4+<9*4>>(SP),- :RETRIEVE ADDRESS OF DESCRIPTORS
    BA AA 00D5 246 :WRK_L_RSLNXT(R10)
    FF26' 30 00D7 247 :BSBW DCL$GETDVAL :GET INPUT FILE DESCRIPTOR VALUES
    7E 51 7D 00DA 248 :MOVQ R1,-(SP) :SAVE INPUT FILESPEC
    54 D4 00DD 249 :CLRL R4 :ASSUME NO OUTPUT FILESPEC
    FF1E' 30 00DF 250 :BSBW DCL$GETDVAL :GET OUTPUT FILESPEC
    03 50 E9 00E2 251 :BLBC R0,65$ :IF NONE, PASS IN NULL FILESPEC
    54 51 7D 00E5 252 :MOVQ R1,R4 :SET OUTPUT FILESPEC ARGUMENT
    52 8E 7D 00E8 253 65$: MOVQ (SP)+,R2 :SET INPUT FILESPEC ARGUMENT
    51 D4 00EB 254 :CLRL R1 :SIGNAL ALL RMS ERRORS
    0049 30 00ED 255 :BSBW DCL$PUSHPROC :PUSH PROCEDURE ONTO INDIRECT STACK
    09 50 E9 00F0 256 :BLBC R0,80$ :BRANCH IF ERROR DETECTED
    00F3 257
    00F3 258 :
    00F3 259 : CREATE SYMBOLS P1-P8
    00F3 260
```


INDIRECT
V04-000

D 15
- INDIRECT FILE MANIPULATION ROUTINES
STACK INDIRECT FILE

15-SEP-1984 23:55:59
4-SEP-1984 23:41:10

VAX/VMS Macro V04-00
[DCL.SRC]INDIRECT.MAR;1

Page 6
(2)

58	56	6E	D0	00F3	261	MOVL	(SP),R6	:GET NUMBER OF SYMBOL DEFINITIONS
	04	AE	9E	00F6	262	MOVAB	4(SP),R8	:GET ADDRESS OF VALUE DESCRIPTORS
		09	10	00FA	263	BSBB	DCL\$DEFINE_P1_TO_P8	:DEFINE P1 THROUGH P8
				00FC	264			
				00FC	265			
				00FC	266			
				00FC	267			
				00FC	268			
5E	44	AE	9E	00FC	268	80\$:	MOVAB <SYMBOLS*8+4>(SP),SP	:DEALLOCATE SYMBOL DESCRIPTOR STORAGE
		50	DD	0100	269		PUSHL R0	:SAVE FINAL STATUS
	024A	31	0102	0102	270		BRW STKXIT	:

```
0105 272 .SBTTL DEFINE SYMBOLS P1-P8
0105 273 :+
0105 274 : DCL$DEFINE_P1_TO_P8 - DEFINE SYMBOLS P1-P8
0105 275 :
0105 276 : THIS ROUTINE IS CALLED TO DEFINE THE LOCAL SYMBOLS P1-P8.
0105 277 :
0105 278 : INPUTS:
0105 279 :
0105 280 : R6 = NUMBER OF SYMBOLS THAT HAVE ASSIGNED VALUES
0105 281 : R8 = ADDRESS OF LIST OF Pn VALUE DESCRIPTORS
0105 282 :
0105 283 : OUTPUTS:
0105 284 :
0105 285 : R1-R8 TRASHED
0105 286 :
0105 287 :-
0105 288
0105 289 DCL$DEFINE P1 TO P8::
7E 3050 8F 3C 0105 290 MOVZWC #A/PO/,-(SP) ;CREATE PROTOTYPE OF GENERATED SYMBO
57 08 D0 010A 291 MOVL #SYMBOLS,R7 ;SET NUMBER OF SYMBOLS TO GENERATE
51 56 D4 010D 292 10$: CLRL R1 ;ASSUME NO MORE SYMBOLS DEFINED
51 03 D7 010F 293 DECL R6 ;ARE THERE ANY MORE TO DEFINE
51 88 19 0111 294 BLSS 20$ ;BR IF NO - DEFINE AS NULL STRING
01 AE 96 0116 295 MOVQ (R8)+,R1 ;GET VALUE DESCRIPTOR
53 02 D0 0119 296 20$: INCB 1(SP) ;INCREMENT SYMBOL NUMBER
54 6E 9E 011C 297 MOVL #2,R3 ;SET LENGTH OF SYMBOL NAME
55 38 AB 9E 011F 298 MOVAB (SP),R4 ;SET ADDRESS OF SYMBOL NAME
50 00 D0 0123 299 MOVAB PRC 0 LOCAL(R11),R5 ;GET ADDRESS OF LOCAL SYMBOL TABLE L
FED7' 30 0126 300 MOVL #SYM R STRING,R0 ;SET SYMBOL TYPE IS STRING
OA 50 E9 0129 301 BSBW DCL$ALCOCSYM ;ALLOCATE AND INSERT SYMBOL TABLE EN
DE 57 F5 012C 302 BLBC R0,90$ ;IF LBC ALLOCATION FAILURE
8E D5 012F 303 SOBGTR R7,10$ ;ANY MORE SYMBOL TO PROCESS?
05 0136 304 STATUS NORMAL ;SET NORMAL COMPLETION STATUS
05 0138 305 90$: TSTL (SP)+ ;RESTORE THE STACK
306 RSB ;RETURN
```

```
0139 308 .SBTTL PUSH PROCEDURE ONTO INDIRECT STACK
0139 309
0139 310 :+ DCL$PUSHPROC - PUSH PROCEDURE ONTO INDIRECT STACK
0139 311
0139 312 THIS ROUTINE IS CALLED TO INITIALIZE A NEW INDIRECT FRAME
0139 313 ON THE INDIRECT PROCEDURE STACK.
0139 314
0139 315 INPUTS:
0139 316
0139 317 R1 = 1 IF RMS ERRORS SHOULD NOT BE SIGNALLED, ELSE 0
0139 318 R2/R3 = DESCRIPTOR OF INPUT FILESPEC
0139 319 R4/R5 = DESCRIPTOR OF OUTPUT FILESPEC
0139 320 R11 = ADDRESS OF PROCESS WORK AREA
0139 321
0139 322 OUTPUTS:
0139 323
0139 324 R0 = STATUS (NOT SIGNALLED)
0139 325
0139 326 :-
0139 327
0139 328 DCL$PUSHPROC::
0139 329 PUSH R1,R2,R3,R4,R5,R6,R7,R8,R9,AP>
0139 330 MOVL PRC_L_STACKPT(R11),R6 :GET CURRENT INDIRECT STACK POINTER
0139 331 MOVAB -IDF_R_LENGTH(R6),R8 :CALCULATE NEW INDIRECT STACK POINTER
0139 332 CMPL R8,PRC_L_STACKLM(R11) :INDIRECT STACK OVERFLOW?
0139 333 BGEQU 2$, :BRANCH IF OK
0139 334 STATUS STKOVF :SET INDIRECT STACK OVERFLOW
0139 335 POP R1,R2,R3,R4,R5,R6,R7,R8,R9,AP>
0139 336 RSB
0139 337
0139 338 :
0139 339 : ALLOCATE ROOM IN SYMBOL TABLE FOR RESULTANT NAME STRING PRIOR TO CLOSING
0139 340 : CURRENT INDIRECT FRAME.
0139 341 :
0139 342 :
0139 343 2$: MOVL #<<<NAMSC_MAXRSS+1>+7>>>8^C<7>>>,R1 :SET MAXIMUM SIZE OF
0139 344 :RESULTANT NAME STRING
0139 345 BSBW DCL$ALLDYNMEM :ALLOCATE ROOM IN THE SYMBOL TABLE
0139 346 BLBS R0,3$ :BR IF NO ALLOCATION ERROR
0139 347 STATUS SYMOVF :INDICATE NO MORE ROOM IN SYM TAB
0139 348 BRB 80$ :JUST EXIT
0139 349 3$: MOVL R2,R9 :SAVE ADDRESS OF ALLOCATED BLOCK
0139 350
0139 351 :
0139 352 : THE NEW INDIRECT FILE FRAME IS FORMED ON THE STACK AND LINKED TO ANY
0139 353 : PREVIOUS FRAMES. THE STACK OVERFLOW CHECK HAS BEENY PERFORMED AT THIS POINT
0139 354 :
0139 355 :
0139 356 INCL PRC_L_INDEPTH(R11) : SET NEW INSTACK STACK DEPTH
0139 357 INCL PRC_L_INDCLOCK(R11) : INCREMENT TOTAL STACKS OR UNSTACKS
0139 358 MOVL R8,PRC_L_STACKPT(R11) : ALLOCATE NEW STACK FRAME
0139 359 MOVL PRC_L_IDFLNK(R11),R6 : GET ADDRESS OF CURRENT INDIRECT FRAME
0139 360 MOVL R6,IDF_L_LNK(R8) : LINK NEW FRAME INTO TOP OF
0139 361 MOVAL IDF_L_LNK(R8), - : INDIRECT FILE FRAME LIST
0139 362 PRC_L_IDFLNK(R11)
0139 363 :
0139 364 : R6 = Pointer to current stack frame
```

56 13FE 8F BB 0139 329
58 00A0 CB D0 013D 330
58 BC A6 9E 0142 331
00A4 CB 58 D1 0146 332
OC 1E 0148 333
13FE 8F BA 014D 334
05 0154 335
0158 336
0159 337
0159 338
0159 339
0159 340
0159 341
0159 342
51 00000100 8F D0 0159 343
FE9D 30 0160 344
09 50 E8 0160 345
E5 11 0163 346
59 52 D0 0166 347
016D 348
016F 349
0172 350
0172 351
0172 352
0172 353
0172 354
0172 355
5C AB D6 0172 356
7C AB D6 0175 357
00A0 CB 58 D0 0178 358
56 00BC CB D0 017D 359
68 56 D0 0182 360
00BC CB 68 DE 0185 361
018A 362
018A 363
018A 364


```
018A 365 : R8 = Pointer to new stack frame
018A 366 :
10 5C 1C AB D0 018A 367 :      MOVL   PRC_L_INDFAB(R11),AP      :GET ADDRESS OF INDIRECT FAB
18 A6 38 AB 7D 018E 368 :      MOVQ   PRC_Q_LOCAL(R11),IDF_Q_LOCAL(R6) :SAVE LOCAL SYMBOL TABLE LISTHEAD
06 A6 30 AB 7D 0193 369 :      MOVQ   PRC_Q_LABEL(R11),IDF_Q_LABEL(R6) :SAVE LABEL SYMBOL TABLE LISTHEAD
08 A6 6A AB B0 0198 370 :      MOVW   PRC_W_ONLEVEL(R11),IDF_W_ONLEVEL(R6) :SAVE ON ERROR LEVEL NUMBER
50 A6 6C AB D0 019D 371 :      MOVL   PRC_L_ONERROR(R11),IDF_L_ONERROR(R6) :SAVE ON ERROR COMMAND TEXT
50 60 38 AB 9E 01A2 372 :      MOVAB   PRC_Q_LOCAL(R11),R0      :GET ADDRESS OF LOCAL TABLE LISTHEAD
60 80 50 D0 01A6 373 :      MOVL   R0,TR0      :SET ADDRESS OF LISTHEAD AS FORWARD LINK
80 80 D0 01A9 374 :      MOVL   (R0)+,(R0)+      :SET ADDRESS OF LISTHEAD AS BACKWARD LINK
50 30 AB 9E 01AC 375 :      MOVAB   PRC_Q_LABEL(R11),R0      :GET ADDRESS OF LABEL TABLE LISTHEAD
60 50 D0 01B0 376 :      MOVL   R0,TR0      :SET ADDRESS OF LISTHEAD AS FORWARD LINK
80 80 D0 01B3 377 :      MOVL   (R0)+,(R0)+      :SET ADDRESS OF LISTHEAD AS BACKWARD LINK
6A AB 0202 8F B0 01B6 378 :      CLRL   PRC_L_ONERROR(R11)      :CLEAR ADDRESS OF ON ERROR COMMAND TEXT
60 A6 00B8 CB D0 01B9 379 :      MOVW   #208!2,PRC_W_ONLEVEL(R11) :RESET ON ERROR LEVEL TO ERROR
00B8 CB 0000 CF 9E 01C5 381 :      BEQL   $$      :BR IF THERE WAS NONE
5E AB 01 B0 01C7 382 :      MOVAB   W^DCLST_DEFONTXT,PRC_L_ONCTLY(R11) :SET DEFAULT FOR NEXT LEVEL
64 AB D4 01CE 383 5$:      MOVW   #10IDF_V_INPOPN,IDF_Q_FLAG(R8) :SET INPUT FILE OPEN FLAG
01D2 384 :      :ASSUME FILE IS OPENED LOCALLY
01D2 385 :      CLRL   IDF_L_SEARCHCTX(R8)      :INITIALIZE F$SEARCH CONTEXT LIST
01D5 386 :
01D5 387 :
01D5 388 :      :CLOSE INPUT FILE FROM PREVIOUS INDIRECT LEVEL AND REMEMBER THE CURRENT
01D5 389 :      :POSITION IN THE FILE, SO THAT ON RETURN, WE CAN RESET THE POSITION.
01D5 390 :
52 14 AB D0 01D5 391 :      MOVL   PRC_L_INDIRPRAB(R11),R2 :SET CURRENT INDIRECT RAB POINTER
08 AB 52 D1 01D9 392 :      CMPL   R2,PRC_L_INPRAB(R11) :IS THIS THE PRIMARY INPUT STREAM?
1E 18 A2 1C E1 01DD 393 :      BEQL   7$      :BR IF YES-THAT NEVER GETS CLOSED
58 A6 01 AE 01DF 394 :      BBC    #DEV$V_RND,RAB$SL_CTX(R2) :6$ :SKIP IF NOT A DISK FILE
0000 8F 50 B1 01E4 395 :      MNEGW   #1,IDF_W_INPRFA(R6) :ASSUME END OF FILE
58 A6 10 A2 D0 01E8 396 :      $FIND   RAB=(R2) :GET THE CURRENT RECORD POSITION (IT
5C A6 14 A2 B0 01F1 397 :      CMPW   R0,#RMSS_EOF&*XFFFF :MAY HAVE BEEN ADVANCED BY AN INDIRECT
02 AC 04 A6 B0 01F6 398 :      BEQL   6$      :ACCESSOR SINCE OUR LAST $GET).
0202 400 :      MOVL   RAB$W_RFA(R2),IDF_W_INPRFA(R6) :SAVE RECORD POSITION IN FILE
0207 401 6$:      MOVW   RAB$W_RFA+4(R2),IDF_W_INPRFA+4(R6)
0210 402 :      MOVW   IDF_W_INPIFI(R6),FAB$W_IFI(AP) :SET INTERNAL FILE IDENTIFICATION
0210 403 :      $CLOSE   FAB=(AP)
0210 404 :
0210 405 :      :OPEN INPUT PROCEDURE FILE
0210 406 :
04 AB B4 0210 407 7$:      CLRW   IDF_W_INPIFI(R8) :CLEAR INPUT FILE INTERNAL INDEX
02 AC B4 0213 408 :      CLRW   FAB$W_IFI(AP) :CLEAR INTERNAL FILE INDEX
0216 409 :
51 04 AE 7D 0216 410 :      MOVQ   4(SP),R1 :GET INPUT FILESPEC (R2/R3 ON ENTRY)
34 AC 51 90 021A 411 :      MOVB   R1,FAB$B_FNS(AP) :SET SIZE OF FILE NAME STRING
2C AC 52 D0 021E 412 :      MOVL   R2,FAB$B_FNA(AP) :SET ADDRESS OF FILE NAME STRING
35 AC 04 90 0222 413 :      MOVB   #4,FAB$B_DNS(AP) :SET SIZE OF DEFAULT NAME STRING
30 AC FDD6 CF 9E 0226 414 :      MOVAB   INPFILE,FAB$B_DNA(AP) :SET ADDRESS OF DEFAULT NAME STRING
57 28 AC D0 022C 415 :
57 28 AC D0 022C 416 :      MOVL   FAB$B_NAM(AP),R7 :GET ADDRESS OF INDIRECT NAME BLOCK
69 FF 8F 90 0230 417 :
68 AB 59 D0 0230 418 :      MOVB   #255,(R9) :STORE LENGTH OF BUFFER (IN SYM TAB)
FF 8F 98 0234 419 :      MOVL   R9,IDF_L_FILENAME(R8) :STORE ADDRESS OF BUFFER (IN SYM TAB)
FF 8F 98 0238 420 :      ASSUME   NAMS$B_RSC EQ NAMS$B_RSS+1
FF 8F 98 0238 421 :      MOVZBW   #NAMS$C_MAXRSS,- :SAVE THE SIZE OF THE BUFFER
```

```
04 A7 02 A7 9E 023B 422      NAM$B_RSS(R7)      ;(NOTE, NOT THE ALLOCATED SIZE)
01 A9 01 A9 9E 023D 423      1(R9),NAM$B_RSA(R7) ;SAVE THE ADDRESS OF THE BUFFER
      FF 8F 9B 0242 424      ASSUME NAM$B_RSL EQ NAM$B_RSS+1
      0A A7 9B 0242 425      MOVZBW #NAM$C_MAXRSS,-      ;SET UP EXPANDED STRING TOO
      04 A7 D0 0245 426      NAM$B_ESS(R7)      ;
      0C A7 D0 0247 427      NAM$B_RSA(R7),-      ;
      0C A7 D0 024A 428      NAM$B_ESA(R7)      ;
      08 A7 01 90 024C 429
16 AC 80 8F 90 0250 430      MOVB #NAM$M_PWD,NAM$B_NOP(R7);DISABLE PASSWORD MASKING
000C0000 8F D0 0255 431      MOVB #FAB$M_EXE,FAB$B_FAC(AP);SET FILE ACCESS TYPE
      04 AC D0 0258 432      MOVBL #FAB$M_INP,FAB$M_PPF,- ;SET FILE OPEN OPTIONS
      1E AC 04 90 025D 433      FAB$B_FOP(AP)      ;
      1F AC 03 90 0261 434      MOVBL #FAB$M_PRN,FAB$B_RAT(AP);SET CARRIAGE CONTROL
      17 AC 94 0265 435      MOVBL #FAB$C_VFC,FAB$B_RFM(AP);SET VERT. FORMS CONTROL
      50 5C D0 0268 436      CLRB FAB$B_SHR(AP)      ;CLEAR FILE SHARING OPTIONS
      51 04 D0 026B 437      MOVBL AP,R0      ;ADDRESS OF FAB
      03 6E E9 026B 438      MOVBL #4,R1      ;ASSUME OPEN WITH ERROR REPORTING
      51 02 E9 026E 439      BLBC (SP),8$      ;IF ERROR REPORTING DISABLED,
      FD89 30 C8 0271 440      BLSL #2,R1      ;DO OPEN WITHOUT ERROR REPORTING
      3D 50 E9 0274 441      BSBW DCL$OPEN CREATE ;OPEN INDIRECT INPUT FILE
      3D 50 E9 0277 442      CLRBIT NAM$V_PWD,NAM$B_NOP(R7);UNCONDITIONALLY REENABLE PASSWORD MASKING
      3D 50 E9 027B 443      BLBC R0,9$      ;IF LBC OPEN FAILURE
      04 A8 02 AC B0 027E 444
56 00F4 CC 9E 0283 445      MOVW FAB$W_IFI(AP),IDF_W_INPIFI(R8);SAVE INPUT FILE INTERNAL INDEX
18 A6 40 AC D0 0288 446      MOVAB PRD_G_ALTINPRAB(AP),R6 ;GET ALTERNATE INPUT RAB
OC A8 18 A6 D0 028D 447      MOVBL FAB$B_DEV(AP),RAB$B_CTX(R6);SAVE DEVICE CHARACTERISTICS
      11 E1 0292 448      MOVBL RAB$B_CTX(R6),IDF_L_INPRABCTX(R8);AND A COPY IN THE STACK FRAME
      04 34 A7 01 0294 449      BBC #NAM$V_NODE,- ;BRANCH IF NOT A REMOTE OPEN
      04 34 A7 01 0297 450      NAM$B_FNB(R7),10$      ;
      04 34 A7 01 0297 451      SETBIT IDF_V_REMOTE,- ;SET REMOTE OPEN FLAG
      04 34 A7 01 0297 452      IDF_W_FLAG(R8)      ;
      04 34 A7 01 0298 453
      04 34 A7 01 0298 454
      04 34 A7 01 0298 455
      04 34 A7 01 0298 456
      04 34 A7 01 0298 457
      04 34 A7 01 0298 458
      04 34 A7 01 029F 459
      04 34 A7 01 02A1 460
      04 34 A7 01 02A4 461
      04 34 A7 01 02A8 462
      04 34 A7 01 02A8 463
      14 A7 1C 28 02A8 464      10$: CLRBIT IDF_V_INPCCL,IDF_B_OUTFLAGS(R8);CLEAR CONCEALED BIT IN IDF
      3C A6 5C D0 02AC 465      BBC #NAM$V_CNCL_DEV,- ;IS DEVICE CONCEALED?
      4A 50 E9 02B2 466      NAM$B_FNB(R7),11$      ;
      14 AB 56 D0 02B8 467      SETBIT IDF_V_INPCCL,IDF_B_OUTFLAGS(R8);SET CONCEALED BIT IN IDF
      0272 30 02BE 468      ASSUME IDF_W_INPFID EQ IDF_T_INPDVI+16 ;
      0272 30 02C3 469      ASSUME IDF_W_INPDID EQ IDF_W_INPFID+6 ;
      0272 30 02C7 470      11$: MOVBL #28,NAM$T_DVI(R7),- ;COPY FILE INFORMATION
      0272 30 02CA 471      IDF_T_INPDVI(R8) ;INTO INDIRECT STACK FRAME
      0272 30 02CA 472      MOVBL AP,RAB$B_FAB(R6) ;LINK FAB TO RAB
      0272 30 02CA 473      $CONNECT RAB=(R6) ;CONNECT TO NEW INPUT
      0272 30 02CA 474      BLBC R0,50$ ;IF LBC CONNECT FAILURE
      0272 30 02CA 475      CLRBIT RAB$V_PPF_IND,RAB$W_ISI(R6);MAKE SURE INDIRECT FLAG IS CLEAR
      0272 30 02CE 476      MOVBL R6,PRD_L_INDINPRAB(R11);SET INDIRECT INPUT RAB
      0272 30 02D2 477      BSBW SAV_EXE_ONLY ;SAVE VER. FLAGS IF EXE-ONLY PROCEDURE.
      0272 30 02D2 478
      0272 30 02D2 479
      0272 30 02D2 480
      0272 30 02D2 481
      0272 30 02D2 482
      0272 30 02D2 483
      0272 30 02D2 484
      0272 30 02D2 485
      0272 30 02D2 486
      0272 30 02D2 487
      0272 30 02D2 488
      0272 30 02D2 489
      0272 30 02D2 490
      0272 30 02D2 491
      0272 30 02D2 492
      0272 30 02D2 493
      0272 30 02D2 494
      0272 30 02D2 495
      0272 30 02D2 496
      0272 30 02D2 497
      0272 30 02D2 498
      0272 30 02D2 499
      0272 30 02D2 500
      0272 30 02D2 501
      0272 30 02D2 502
      0272 30 02D2 503
      0272 30 02D2 504
      0272 30 02D2 505
      0272 30 02D2 506
      0272 30 02D2 507
      0272 30 02D2 508
      0272 30 02D2 509
      0272 30 02D2 510
      0272 30 02D2 511
      0272 30 02D2 512
      0272 30 02D2 513
      0272 30 02D2 514
      0272 30 02D2 515
      0272 30 02D2 516
      0272 30 02D2 517
      0272 30 02D2 518
      0272 30 02D2 519
      0272 30 02D2 520
      0272 30 02D2 521
      0272 30 02D2 522
      0272 30 02D2 523
      0272 30 02D2 524
      0272 30 02D2 525
      0272 30 02D2 526
      0272 30 02D2 527
      0272 30 02D2 528
      0272 30 02D2 529
      0272 30 02D2 530
      0272 30 02D2 531
      0272 30 02D2 532
      0272 30 02D2 533
      0272 30 02D2 534
      0272 30 02D2 535
      0272 30 02D2 536
      0272 30 02D2 537
      0272 30 02D2 538
      0272 30 02D2 539
      0272 30 02D2 540
      0272 30 02D2 541
      0272 30 02D2 542
      0272 30 02D2 543
      0272 30 02D2 544
      0272 30 02D2 545
      0272 30 02D2 546
      0272 30 02D2 547
      0272 30 02D2 548
      0272 30 02D2 549
      0272 30 02D2 550
      0272 30 02D2 551
      0272 30 02D2 552
      0272 30 02D2 553
      0272 30 02D2 554
      0272 30 02D2 555
      0272 30 02D2 556
      0272 30 02D2 557
      0272 30 02D2 558
      0272 30 02D2 559
      0272 30 02D2 560
      0272 30 02D2 561
      0272 30 02D2 562
      0272 30 02D2 563
      0272 30 02D2 564
      0272 30 02D2 565
      0272 30 02D2 566
      0272 30 02D2 567
      0272 30 02D2 568
      0272 30 02D2 569
      0272 30 02D2 570
      0272 30 02D2 571
      0272 30 02D2 572
      0272 30 02D2 573
      0272 30 02D2 574
      0272 30 02D2 575
      0272 30 02D2 576
      0272 30 02D2 577
      0272 30 02D2 578
      0272 30 02D2 579
      0272 30 02D2 580
      0272 30 02D2 581
      0272 30 02D2 582
      0272 30 02D2 583
      0272 30 02D2 584
      0272 30 02D2 585
      0272 30 02D2 586
      0272 30 02D2 587
      0272 30 02D2 588
      0272 30 02D2 589
      0272 30 02D2 590
      0272 30 02D2 591
      0272 30 02D2 592
      0272 30 02D2 593
      0272 30 02D2 594
      0272 30 02D2 595
      0272 30 02D2 596
      0272 30 02D2 597
      0272 30 02D2 598
      0272 30 02D2 599
      0272 30 02D2 600
      0272 30 02D2 601
      0272 30 02D2 602
      0272 30 02D2 603
      0272 30 02D2 604
      0272 30 02D2 605
      0272 30 02D2 606
      0272 30 02D2 607
      0272 30 02D2 608
      0272 30 02D2 609
      0272 30 02D2 610
      0272 30 02D2 611
      0272 30 02D2 612
      0272 30 02D2 613
      0272 30 02D2 614
      0272 30 02D2 615
      0272 30 02D2 616
      0272 30 02D2 617
      0272 30 02D2 618
      0272 30 02D2 619
      0272 30 02D2 620
      0272 30 02D2 621
      0272 30 02D2 622
      0272 30 02D2 623
      0272 30 02D2 624
      0272 30 02D2 625
      0272 30 02D2 626
      0272 30 02D2 627
      0272 30 02D2 628
      0272 30 02D2 629
      0272 30 02D2 630
      0272 30 02D2 631
      0272 30 02D2 632
      0272 30 02D2 633
      0272 30 02D2 634
      0272 30 02D2 635
      0272 30 02D2 636
      0272 30 02D2 637
      0272 30 02D2 638
      0272 30 02D2 639
      0272 30 02D2 640
      0272 30 02D2 641
      0272 30 02D2 642
      0272 30 02D2 643
      0272 30 02D2 644
      0272 30 02D2 645
      0272 30 02D2 646
      0272 30 02D2 647
      0272 30 02D2 648
      0272 30 02D2 649
      0272 30 02D2 650
      0272 30 02D2 651
      0272 30 02D2 652
      0272 30 02D2 653
      0272 30 02D2 654
      0272 30 02D2 655
      0272 30 02D2 656
      0272 30 02D2 657
      0272 30 02D2 658
      0272 30 02D2 659
      0272 30 02D2 660
      0272 30 02D2 661
      0272 30 02D2 662
      0272 30 02D2 663
      0272 30 02D2 664
      0272 30 02D2 665
      0272 30 02D2 666
      0272 30 02D2 667
      0272 30 02D2 668
      0272 30 02D2 669
      0272 30 02D2 670
      0272 30 02D2 671
      0272 30 02D2 672
      0272 30 02D2 673
      0272 30 02D2 674
      0272 30 02D2 675
      0272 30 02D2 676
      0272 30 02D2 677
      0272 30 02D2 678
      0272 30 02D2 679
      0272 30 02D2 680
      0272 30 02D2 681
      0272 30 02D2 682
      0272 30 02D2 683
      0272 30 02D2 684
      0272 30 02D2 685
      0272 30 02D2 686
      0272 30 02D2 687
      0272 30 02D2 688
      0272 30 02D2 689
      0272 30 02D2 690
      0272 30 02D2 691
      0272 30 02D2 692
      0272 30 02D2 693
      0272 30 02D2 694
      0272 30 02D2 695
      0272 30 02D2 696
      0272 30 02D2 697
      0272 30 02D2 698
      0272 30 02D2 699
      0272 30 02D2 700
      0272 30 02D2 701
      0272 30 02D2 702
      0272 30 02D2 703
      0272 30 02D2 704
      0272 30 02D2 705
      0272 30 02D2 706
      0272 30 02D2 707
      0272 30 02D2 708
      0272 30 02D2 709
      0272 30 02D2 710
      0272 30 02D2 711
      0272 30 02D2 712
      0272 30 02D2 713
      0272 30 02D2 714
      0272 30 02D2 715
      0272 30 02D2 716
      0272 30 02D2 717
      0272 30 02D2 718
      0272 30 02D2 719
      0272 30 02D2 720
      0272 30 02D2 721
      0272 30 02D2 722
      0272 30 02D2 723
      0272 30 02D2 724
      0272 30 02D2 725
      0272 30 02D2 726
      0272 30 02D2 727
      0272 30 02D2 728
      0272 30 02D2 729
      0272 30 02D2 730
      0272 30 02D2 731
      0272 30 02D2 732
      0272 30 02D2 733
      0272 30 02D2 734
      0272 30 02D2 735
      0272 30 02D2 736
      0272 30 02D2 737
      0272 30 02D2 738
      0272 30 02D2 739
      0272 30 02D2 740
      0272 30 02D2 741
      0272 30 02D2 742
      0272 30 02D2 743
      0272 30 02D2 744
      0272 30 02D2 745
      0272 30 02D2 746
      0272 30 02D2 747
      0272 30 02D2 748
      0272 30 02D2 749
      0272 30 02D2 750
      0272 30 02D2 751
      0272 30 02D2 752
      0272 30 02D2 753
      0272 30 02D2 754
      0272 30 02D2 755
      0272 30 02D2 756
      0272 30 02D2 757
      0272 30 02D2 758
      0272 30 02D2 759
      0272 30 02D2 760
      0272 30 02D2 761
      0272 30 02D2 762
      0272 30 02D2 763
      0272 30 02D2 764
      0272 30 02D2 765
      0272 30 02D2 766
      0272 30 02D2 767
      0272 30 02D2 768
      0272 30 02D2 769
      0272 30 02D2 770
      0272 30 02D2 771
      0272 30 02D2 772
      0272 30 02D2 773
      0272 30 02D2 774
      0272 30 02D2 775
      0272 30 02D2 776
      0272 30 02D2 777
      0272 30 02D2 778
      0272 30 02D2 779
      0272 30 02D2 780
      0272 30 02D2 781
      0272 30 02D2 782
      0272 30 02D2 783
      0272 30 02D2 784
      0272 30 02D2 785
      0272 30 02D2 786
      0272 30 02D2 787
      0272 30 02D2 788
      0272 30 02D2 789
      0272 30 02D2 790
      0272 30 02D2 791
      0272 30 02D2 792
      0272 30 02D2 793
      0272 30 02D2 794
      0272 30 02D2 795
      0272 30 02D2 796
      0272 30 02D2 797
      0272 30 02D2 798
      0272 30 02D2 799
      0272 30 02D2 800
      0272 30 02D2 801
      0272 30 02D2 802
      0272 30 02D2 803
      0272 30 02D2 804
      0272 30 02D2 805
      0272 30 02D2 806
      0272 30 02D2 807
      0272 30 02D2 808
      0272 30 02D2 809
      0272 30 02D2 810
      0272 30 02D2 811
      0272 30 02D2 812
      0272 30 02D2 813
      0272 30 02D2 814
      0272 30 02D2 815
      0272 30 02D2 816
      0272 30 02D2 817
      0272 30 02D2 818
      0272 30 02D2 819
      0272 30 02D2 820
      0272 30 02D2 821
      0272 30 02D2 822
      0272 30 02D2 823
      0272 30 02D2 824
      0272 30 02D2 825
      0272 30 02D2 826
      0272 30 02D2 827
      0272 30 02D2 828
      0272 30 02D2 829
      0272 30 02D2 830
      0272 30 02D2 831
      0272 30 02D2 832
      0272 30 02D2 833
      0272 30 02D2 834
      0272 30 02D2 835
      0272 30 02D2 836
      0272 30 02D2 837
      0272 30 02D2 838
      0272 30 02D2 839
      0272 30 02D2 840
      0272 30 02D2 841
      0272 30 02D2 842
      0272 30 02D2 843
      0272 30 02D2 844
      0272 30 02D2 845
      0272 30 02D2 846
      0272 30 02D2 847
      0272 30 02D2 848
      0272 30 02D2 849
      0272 30 02D2 850
      0272 30 02D2 851
      0272 30 02D2 852
      0272 30 02D2 853
      0272 30 02D2 854
      0272 30 02D2 855
      0272 30 02D2 856
      0272 30 02D2 857
      0272 30 02D2 858
      0272 30 02D2 859
      0272 30 02D2 860
      0272 30 02D2 861
      0272 30 02D2 862
      0272 30 02D2 863
      0272 30 02D2 864
      0272 30 02D2 865
      0272 30 02D2 866
      0272 30 02D2 867
      0272 30 02D2 868
      0272 30 02D2 869
      0272 30 02D2 870
      0272 30 02D2 871
      0272 30 02D2 872
      0272 30 02D2 873
      0272 30 02D2 874
      0272 30 02D2 875
      0272 30 02D2 876
      0272 30 02D2 877
      0272 30 02D2 878
      0272 30 02D2 879
      0272 30 02D2 880
      0272 30 02D2 881
      0272 30 02D2 882
      0272 30 02D2 883
      0272 30 02D2 884
      0272 30 02D2 885
      0272 30 02D2 886
      0272 30 02D2 887
      0272 30 02D2 888
      0272 30 02D2 889
      0272 30 02D2 890
      0272 30 02D2 891
      0272 30 02D2 892
      0272 30 02D2 893
      0272 30 02D2 894
      0272 30 02D2 895
      0272 30 02D2 896
      0272 30 02D2 897
      0272 30 02D2 898
      0272 30 02D2 899
      0272 30 02D2 900
      0272 30 02D2 901
      0272 30 02D2 902
      0272 30 02D2 903
      0272 30 02D2 904
      0272 30 02D2 905
      0272 30 02D2 906
      0272 30 02D2 907
      0272 30 02D2 908
      0272 30 02D2 909
      0272 30 02D2 910
      0272 30 02D2 911
      0272 30 02D2 912
      0272 30 02D2 913
      0272 30 02D2 914
      0272 30 02D2 915
      0272 30 02D2 916
      0272 30 02D2 917
      0272 30 02D2 918
      0272 30 02D2 919
      0272 30 02D2 920
      0272 30 02D2 921
      0272 30 02D2 922
      0272 30 02D2 923
      0272 30 02D2 924
      0272 30 02D2 925
      0272 30 02D2 926
      0272 30 02D2 927
      0272 30 02D2 928
      0272 30 02D2 929
      0272 30 02D2 930
      0272 30 02D2 931
      0272 30 02D2 932
      0272 30 02D2 933
      0272 30 02D2 934
      0272 30 02D2 935
      0272 30 02D2 936
      0272 30 02D2 937
      0272 30 02D2 938
      0272 30 02D2 939
      0272 30 02D2 940
      0272 30 02D2 941
      0272 30 02D2 942
      0272 30 02D2 943
      0272 30 02D2 944
      0272 30 02D2 945
      0272 30 02D2 946
      0272 30 02D2 947
      0272 30 02D2 948
      0272 30 02D2 949
      0272 30 02D2 950
      0272 30 02D2 951
      0272 30 02D2 952
      0272 30 02D2 953
      0272 30 02D2 954
      0272 30 02D2 955
      0272 30 02D2 956
      0272 30 02D2 957
      0272 30 02D2 958
      0272 30 02D2 959
      0272 30 02D2 960
      0272 30 02D2 961
      0272 30 02D2 962
      0272 30 02D2 963
      0272 30 02D2 964
      0272 30 02D2 965
      0272 30 02D2 966
      0272 30 02D2 967
      0272 30 02D2 968
      0272 30 02D2 969
      0272 30 02D2 970
      0272 30 02D2 971
      0272 30 02D2 972
      0272 30 02D2 973
      0272 30 02D2 974
      0272 30 02D2 975
      0272 30 02D2 976
      0272 30 02D2 977
      0272 30 02D2 978
      0272 30 02D2 979
      0272 30 02D2 980
      0272 30 02D2 981
      0272 30 02D2 982
      0272 30 02D2 983
      0272 30 02D2 984
      0272 30 02D2 985
      0272 30 02D2 986
      0272 30 02D2 987
      0272 30 02D2 988
      0272 30 02D2 989
      0272 30 02D2 990
      0272 30 02D2 991
      0272 30 02D2 992
      0272 30 02D2 993
      0272 30 02D2 994
      0272 30 02D2 995
      0272 30 02D2 996
      0272 30 02D2 997
      0272 30 02D2 998
      0272 30 02D2 999
      0272 30 02D2 1000
      0272 30 02D2 1001
      0272 30 02D2 1002
      0272 30 02D2 1003
      0272 30 02D2 1004
      0272 30 02D2 1005
      0272 30 02D2 1006
      0272 30 02D2 1007
      0272 30 02D2 1008
      0272 30 02D2 1009
      0272 30 02D2 1010
      0272 30 02D2 1011
      0272 30 02D2 1012
      0272 30 02D2 1013
      0272 30 02D2 1014
     
```

```
FD28' 30 02D5 479 BSBW DCL$OPEN_OUTPUT ;CONDITIONALLY OPEN SYS$OUTPUT
51 8ED0 02D8 480 POPL R1 ;RESTORE LENGTH OF INPUT FILE NAME
2A 50 E9 02DB 481 BLBC R0,50$ ;RETURN ANY ERRORS
02DE 482
02DE 483
02DE 484
02DE 485
02DE 486
50 68 A8 D0 02E2 487 MOVL IDF L FILENAME(RB),R0 ;GET ADDRESS OF BUFFER
60 51 90 02E2 487 MOVB R1,(R0) ;SAVE FILE NAME LENGTH IN FIRST BYTE OF BUFF
51 08 C0 02E3 488 ADDL #8,R1 ;ROUND UP SIZE TO QUADWORD BOUNDARY(INCLUDE
51 07 CA 02E8 489 BICL #7,R1 ;TRUNCATE DOWN SIZE TO QUADWORD BOUNDARY
50 51 C0 02EB 490 ADDL R1,R0 ;CALCULATE ADDRESS OF UNUSED BUFFER
51 00000100 BF 51 C3 02EE 491 SUBL3 R1,#<<<NAM$C_MAXRSS+1>>+7>B^C<7>>,R1 ;CALCULATE SIZE OF UNUSED BUFFER
03 13 02F6 492 BEQL 40$ ;DON'T DEALLOCATE IF NO UNUSED BUFFER
FD05' 30 02F8 493 BSBW DCL$DEADYNMEM ;DEALLOCATE UNUSED BUFFER
02FB 494
02FB 495
02FB 496
02FB 497
FD02' 30 02FB 498 40$: BSBW DCL$CREATE_IO ;CREATE LOGICAL NAMES FOR 'INPUT' AND 'OUTPU
FE4C 31 02FE 499 STATUS NORMAL
0305 500 BRW 80$ ;EXIT WITH SUCCESS
0308 501
0308 502
0308 503
0308 504
02 A7 B4 0308 505 50$: CLRW NAM$B_RSS(R7) ;INVALIDATE RESULTANT STRINGS
0A A7 B4 0308 506 CLRW NAM$B_ESS(R7) ;INVALIDATE EXPANDED STRINGS
50 DD 030E 507 PUSHL R0 ;SAVE ERROR/STATUS VALUE
0080 30 0310 508 BSBW UNSTACK ;UNSTACK PREVIOUS INDIRECT FILE
50 8ED0 0313 509 POPL R0 ;RETRIEVE ERROR/STATUS VALUE
FE3B 31 0316 510 BRW 80$ ;EXIT WITH STATUS
```



```
0319 512 .SBTTL UNSTACK INDIRECT FILE SPECIFICATION
0319 513
0319 514 + DCL$UNSTACK - UNSTACK INDIRECT FILE SPECIFICATION
0319 515
0319 516 THIS ROUTINE IS CALLED TO CLOSE THE CURRENT INDIRECT FILE AND TO UNSTACK THE
0319 517 PREVIOUS SPECIFICATION.
0319 518
0319 519 INPUTS:
0319 520
0319 521 NONE.
0319 522
0319 523 OUTPUTS:
0319 524
0319 525 THE CURRENT INDIRECT FILE IS CLOSED AND ALL LOCAL SYMBOLS FOR THE LEVEL
0319 526 ARE DEALLOCATED. THE PREVIOUS INDIRECT FILE IS THEN UNSTACKED AND REOPENED.
0319 527
0319 528 R0 LOW BIT CLEAR INDICATES UNSUCCESSFUL COMPLETION.
0319 529
0319 530 R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
0319 531
0319 532 ALL ERRORS ARE SIGNALLED BEFORE RETURNING TO CALLER.
0319 533
0319 534
0319 535 DCL$UNSTACK::
0319 536 $DELLOG_S TBLFLG=#LOG$C_PROCESS,- ;UNSTACK INDIRECT FILE SPECIFICATION
0319 537 ACMODE=#PSL$C_USER ;DELETE ANY USER DEFINED LOGICAL NAMES.
0326 538 BSBB SETIND ;SETUP INDIRECT PROCESSING
0328 539 PUSHL S^#SS$_NORMAL ;ASSUME NORMAL COMPLETION
032A 540 BBCC #PRC_V-GOTO,PRC_W_FLAGS(R11),10$ ;IF CLR, NO GOTO IN PROGRESS
032F 541 BSBB DCL$DEALGOTO ;DEALLOCATE GOTO SYMBOL
0332 542 STATUS USGOTO ;SET UNSATISFIED GOTO STATUS
0339 543 MOVL R0,(SP) ;SET COMPLETION STATUS
033C 544 ERRMSG ;OUTPUT ERROR MESSAGE
033F 545 BSBB DCL$SET_STATUS ;GIVE ERROR HANDLER'S A CHANCE
0342 546 10$: BSBB UNSTACK ;UNSTACK TO PREVIOUS LEVEL
0344 547 MOVAB WRK_G_INPBUF-1(R10),- ;SET STARTING ADDRESS OF INPUT
0348 548 WRK_L_CHARPTR(R10) ;BUFFER AS LAST BYTE FETCHED
034B 549 CLR B WRK_G_INPBUF(R10) ;SET EOL AS NEXT BYTE TO FETCH
034F 550 STKXIT: POPR #^MZR0,R1,R2,R3,R4,R5,R6,R7,R8,AP>;RESTORE REGISTERS
0353 551 ;R0=STATUS, R1=ORIGINAL FLAGS
0353 552 POPL WRK_L_RSLNXT(R10) ;RESTORE TOKEN DESCRIPTORS BACK TO
0357 553 POPL WRK_L_RSLEND(R10) ;WHERE THEY WERE WHEN WE STARTED
035B 554 POPL WRK_L_EXPANDPTR(R10) ;RESTORE EXPANSION BUFFER POINTER
0360 555 POPL WRK_L_MARKPTR(R10) ;RESTORE MARKER POINTER
0365 556 ENABLE ;ENABLE CONTROL Y/C AST'S
0367 557 BBS #WRK_V_COMMAND,R1,10$ ;BR IF COMMAND WAS SET
036B 558 BICW #WRK_M_COMMAND,WRK_W_FLAGS(R10) ;CLEAR COMMAND IN PROGRESS
036F 559 10$: RSB ;
0370 560
0370 561
0370 562 SETIND - SETUP INDIRECT
0370 563
0370 564 SAVE THE NON-VOLATILE REGISTERS AND THE COMMAND WORK FLAGS, THEN SET COMMAND
0370 565
0370 566
0370 567 SETIND: POPR #^M<R0> ;GET RETURN PC
0372 568 DISABLE ;DISABLE CONTROL Y/C AST'S
```

INDIRECT
V04-000

K 15
- INDIRECT FILE MANIPULATION ROUTINES
UNSTACK INDIRECT FILE SPECIFICATION

15-SEP-1984 23:55:59 VAX/VMS Macro V04-00
4-SEP-1984 23:41:10 [DCL.SRC]INDIRECT.MAR;1

Page 13
(5)

F48A	CA	DD	0378	569	PUSHL	WRK_L_MARKPTR(R10)	;SAVE CURRENT MARKER POINTER
F486	CA	DD	037C	570	PUSHL	WRK_L_EXPANDPTR(R10)	;SAVE CURRENT EXPANSION BUFFER POINTER
B6	AA	DD	0380	571	PUSHL	WRK_L_RSLEND(R10)	;SAVE CURRENT ENDING TOKEN ADDRESS
BA	AA	DD	0383	572	PUSHL	WRK_L_RSLNXT(R10)	;SAVE CURRENT POSITION IN TOKEN ARRAY
11FC	8F	BB	0386	573	PUSHR	#MZR2,R3,R4,R5,R6,R7,R8,AP>	;SAVE REGISTERS
F0	AA	DD	038A	574	PUSHL	WRK_W_FLAGS(R10)	;SAVE PREVIOUS COMMAND FLAGS
			038D	575	SETBIT	WRK_V_COMMAND,WRK_W_FLAGS(R10)	;SET COMMAND IN PROGRESS
60	17		0391	576	JMP	(R0)	;RETURN TO CALLER

```
0393 578 .SBTTL UNSTACK NEXT INDIRECT FILE
0393 579 :---
0393 580 :
0393 581 UNSTACK - UNSTACK NEXT INDIRECT FILE
0393 582 :
0393 583 THIS ROUTINE IS CALLED TO CLOSE THE CURRENT INDIRECT FILE AND UNSTACK THE
0393 584 CONTEXT INFORMATION FOR THE PREVIOUS LEVEL INDIRECT FILE.
0393 585 :
0393 586 INPUTS:
0393 587 :
0393 588 R11 = ADDRESS OF PROCESS WORK AREA
0393 589 :
0393 590 OUTPUTS:
0393 591 :
0393 592 NONE
0393 593 :
0393 594 RO-R8,AP ARE DESTROYED.
0393 595 :---
0393 596 :
0393 597 UNSTACK:
0393 598 MOVL PRC_L_INDFAB(R11),AP ;UNSTACK INDIRECT FILE
0393 599 MOVL PRC_L_IDFLNK(R11),R8 ;GET ADDRESS OF SCRATCH FAB
039C 600 PUSHL R8 ;GET ADDRESS OF CURRENT INDIRECT FRAME
039E 601 ;SAVE THAT ADDRESS
039E 602 :
039E 603 : CLOSE CURRENT INPUT PROCEDURE FILE
039E 604 :
039E 605 MOVW IDF_W_INPIFI(R8),FAB$W_IFI(AP) ;RESTORE INTERNAL FILE INDEX
03A3 606 $CLOSE FAB$(AP) ;CLOSE INDIRECT INPUT FILE
03AC 607 :
03AC 608 : DEALLOCATE LOCAL SYMBOLS AND LABELS FOR CURRENT LEVEL
03AC 609 :
03AC 610 :
03AC 611 10$: REMQUE @PRC_Q_LOCAL(R11),R3 ;REMOVE NEXT ENTRY FROM LOCAL SYMBOL TABLE
0380 612 BVC 20$ ;IF VC ENTRY REMOVED
0382 613 REMQUE @PRC_Q_LABEL(R11),R3 ;REMOVE NEXT ENTRY FROM LOCAL LABEL TABLE
0386 614 BVS 30$ ;IF VS TABLE EMPTY
0388 615 20$: BSBW DCL$DEALLOCSYM ;DEALLOCATE SYMBOL ENTRY
038B 616 BRB 10$
038D 617 :
038D 618 : DEALLOCATE F$SEARCH CONTEXT BLOCKS FOR CURRENT LEVEL
038D 619 :
038D 620 30$: MOVL IDF_L_SEARCHCTX(R8),R3 ;GET FIRST ENTRY OFF F$SEARCH LIST
03C1 621 BEQL 32$ ;BRANCH IF NONE LEFT
03C3 622 MOVL (R3),IDF_L_SEARCHCTX(R8) ;REMOVE FROM LINKED LIST
03C7 623 MOVN NLA0,FAB$B_FNS+8(R3) ;SET NULL DEVICE NAME
03CD 624 MOVAB NLA0+1,FAB$L_FNA+8(R3)
03D3 625 $PARSE FAB=8(R3) ;TERMINATE SEARCH SEQUENCE
03DD 626 MOVL R3,R0 ;SET ADDRESS OF BLOCK TO DEALLOCATE
03E0 627 MOVL 4(R0),R1 ;GET SIZE OF ENTRY IN BYTES
03E4 628 BSBW DCL$DEADYNMEM ;DEALLOCATE CONTEXT BLOCK
03E7 629 BRB 30$ ;LOOP UNTIL LIST CLEANED OUT
03E9 630 :
03E9 631 :
03E9 632 : DEALLOCATE FILE NAME STRING
03E9 633 :
03E9 634 32$: MOVL IDF_L_FILENAME(R8),R0 ;GET ADDRESS OF ASCII FILENAME
```

5C 1C AB DO
58 00BC CB DO
58 DD

02 AC 04 A8 B0

53 38 BB OF
06 1C
53 30 BB OF
05 1D
FC45' 30
EF 11

53 64 A8 DO
26 13
64 A8 63 DO
3C A3 FC55 CF 90
34 A3 FC50 CF 9E

51 50 53 DO
04 A0 DO
FC19' 30
D4 11

50 68 A8 DO


```
51 60 9A 03ED 635      MOVZBL (R0),R1      ;GET SIZE OF FILENAME
51 08 C0 03F0 636      ADDL #8,R1      ;ROUND UP SIZE TO QUAD BOUNDARY
51 07 CA 03F3 637      BICL #7,R1      ;TRUNCATE SIZE TO QUAD BOUNDARY
FC07 30 03F6 638      BSBW DCL$DEADYNMEM ;DEALLOCATE BUFFER
      03F9 639
      03F9 640
      03F9 641
      03F9 642      : RESET ON CONDITIONS BACK TO DEFAULTS
FC04 30 03F9 643      BSBW DCL$ONRESET ;RESET ON ERROR PARAMETERS
FC01 30 03FC 644      BSBW DCL$ONCTLYRST ;AND THE ON CONTROL Y HANDLER
      03FF 645
      03FF 646      : CHECK IF THE FRAME JUST CLOSED WAS THE FIRST EXE-ONLY FRAME ENCOUNTERED.
      03FF 647      : IF SO, RESTORE VERIFICATION STATE FROM SAVED FLAGS.
      03FF 648
      0178 30 03FF 649      BSBW RES_EXE_ONLY ;CHECK EXE-ONLY PARAMETERS.
      0402 650
      0402 651      : POINT BACK TO THE PREVIOUS INDIRECT FRAME
      0402 652
      00BC CB 68 D0 0402 653      MOVL IDF_L_LNK(R8), - ; UNLINK FRAME FROM INDIRECT LIST
      0407 654      PRC_L_IDFLNK(R11) ; AND RESET FRAME POINTER
      00A0 CB 74 AB 9E 0407 655      MOVAB IDF_K_LENGTH(R8), - ; REMOVE CURRENT INDIRECT FRAME FROM
      040D 656      PRC_L_STACKPT(R11) ; STACK AND RESET STACK POINTER
      5C AB D7 040D 657      DECL PRC_L_INDEPTH(R11) ; SET NEW INDIRECT STACK DEPTH
      7C AB D6 0410 658      INCL PRC_L_INDCLOCK(R11) ; COUNT TOTAL STACKS OR UNSTACKS
      58 00BC CB D0 0413 659      MOVL PRC_L_IDFLNK(R11),R8 ; POINT TO PREVIOUS INDIRECT FRAME
      0418 660
      0418 661
      0418 662      : RESTORE THE SAVED CONTEXT FROM THE PREVIOUS INDIRECT FRAME
      0418 663
      38 AB 10 AB 7D 0418 664      MOVQ IDF_Q_LOCAL(R8),PRC_Q_LOCAL(R11) ;RESTORE LOCAL SYMBOL TABLE LISTHEA
      30 AB 18 AB 7D 041D 665      MOVQ IDF_Q_LABEL(R8),PRC_Q_LABEL(R11) ;RESTORE LOCAL LABEL TABLE LISTHEAD
      6A AB 06 AB B0 0422 666      MOVW IDF_W_ONLEVEL(R8),PRC_W_ONLEVEL(R11) ;RESTORE ON ERROR LEVEL NUMBER
      6C AB 08 AB D0 0427 667      MOVL IDF_L_ONERROR(R8),PRC_L_ONERROR(R11) ;RESTORE ADDRESS OF COMMAND TEX
      00B8 CB 60 AB D0 042C 668      MOVL IDF_L_ONCTLY(R8),PRC_L_ONCTLY(R11) ;AND THE ON CONTROL T HANDLER
      0432 669
      0432 670
      0432 671      : RE-OPEN THE INPUT PROCEDURE FILE ASSOCIATED WITH THE PREVIOUS
      0432 672      : INDIRECT FRAME.
      0432 673
      FFFF 8F 58 AB B1 0432 674      CMPW IDF_W_INPRFA(R8),#*XFFFF ;IS THE INPUT FILE ALREADY AT EOF?
      03 03 12 0438 675      BNEQ 35$ ;NO, THEN BRANCH
      00EA 31 043A 676      BRW 50$ ;YES, THEN DO NOT REOPEN
      08 AB D0 043D 677 35$: MOVL PRC_L_INPRAB(R11), - ;ASSUME RETURNING TO LEVEL ZERO AND-
      14 AB 0440 678      PRC_L_INDINPRAB(R11) ;SET INPUT AS INDIRECT INPUT ALSO
      03 5E AB 00 E0 0442 679      BBS #IDF_V_INPOPEN,IDF_W_FLAG(R8),351$ ;CONTINUE IF NOT GOING TO LEVEL 0
      008A 31 0447 680      BRW 371$ ;SKIP IF GOING TO LEVEL 0
      044A 681
      044A 682 351$: PUSHL #PSL$C SUPER ;PUSH ACCESS MODE
      7E FB8B CF 9E 044C 683      MOVAB SYS_INPUT_NAME+1,-(SP) ;BUILD LOGICAL NAME DESCRIPTOR
      7E FB85 CF 9A 0451 684      MOVZBL SYS_INPUT_NAME,-(SP)
      7E FB8B CF 9E 0456 685      MOVAB LNMSPROCESS+1,-(SP) ;BUILD TABLE NAME DESCRIPTOR
      7E FB85 CF 9A 045B 686      MOVZBL LNMSPROCESS,-(SP)
      51 5E D0 0460 687      MOVL SP,R1 ;SAVE ADDR. OF DESCRIPTORS
      0463 688      $DELLNM S TABNAM=(R1),- ;DELETE SYSS$INPUT
      0463 689      LOGNAM=8(R1),-
      0463 690      ACMODE=16(R1)
      5E 14 C0 0472 691      ADDL #4*5,SP ;CLEAN STACK
```

56	00F4	CC	9E	0475	692		
14	AB	56	DO	0475	693	MOVAB	PRD G,ALTINPRAB(AP),R6 ;GET THE ALTERNATE INPUT RAB
	0C	A8	DO	047A	694	MOVL	R6,PRC_L,INDINPRAB(R11) ;SET THAT IS INDIRECT INPUT RAB
	18	A6	DO	047E	695	MOVL	IDF_L,INPRABCTX(R8),- ;RESTORE STACKED DEVICE CHARACTERISTICS-
57	28	AC	DO	0481	696		RABSL_CTX(R6) ;VALUE FROM STACK FRAME
				0483	697	MOVL	FABSL_NAM(AP),R7 ;ADDRESS OF NAME BLOCK
				0487	698	ASSUME	IDF_W_INPFID EQ IDF_T_INPDVI+16
				0487	699	ASSUME	IDF_W_INPDID EQ IDF_W_INPFID+6
3C	A8	1C	28	0487	700	MOVC	#28,IDF_T_INPDVI(R8),- ;COPY PREVIOUS INPUT DEVICE,FILE AND-
	14	A7		048B	701		NAMST_DVIT(R7) ;DIRECTORY ID'S INTO NAME BLOCK
				048D	702	ASSUME	NAMSB_DEV EQ NAMSB_NODE+1
				048D	703	ASSUME	NAMSB_DIR EQ NAMSB_DEV+1
				048D	704	ASSUME	NAMSB_NAME EQ NAMSB_DIR+1
				048D	705	ASSUME	NAMSB_TYPE EQ NAMSB_NAME+1
				048D	706	ASSUME	NAMSB_VER EQ NAMSB_TYPE+1
	38	A7	D4	048D	707	CLRL	NAMSB_NODE(R7) ;INIT. FILE SPEC. SIZE FIELDS BEFORE
				0490	708		;REUSING NAM BLOCK.
	3C	A7	B4	0490	709	CLRW	NAMSB_TYPE(R7)
				0493	710		
				0493	711	ASSUME	NAMSB_RSL EQ NAMSB_RSS+1
				0493	712	ASSUME	NAMSB_ESL EQ NAMSB_ESS+1
	02	A7	B4	0493	713	CLRW	NAMSB_RSS(R7) ;SET RESULT RESULTANT AND EXPANDED
	0A	A7	B4	0496	714	CLRW	NAMSB_ESS(R7) ;STRING SIZES TO NULL SO THAT THE
				0499	715		;RSA AND ESA WON'T BE USED.
				0499	716		
16	AC	82	BF	90	0499	MOVB	#FABSM_EXE!FABSM_GET,FABSB_FAC(AP) ;SET FILE ACCESS TYPE
	010C0000	BF	DO	049E	718	MOVL	#FABSM_INP!FABSM_PPF!FABSM_NAM,- ;SET FILE OPEN OPTIONS
		04	AC		04A4		FABSL_FOP(AP)
		34	AC	94	04A6	CLRB	FABSB_FNS(AP) ;REMOVE RESIDUAL FILE NAME SIZE
		17	AC	94	04A9	CLRB	FABSB_SHR(AP) ;CLEAR FILE SHARING OPTIONS
		02	AC	B4	04AC	CLRW	FABSW_IFI(AP) ;CLEAR INPUT IFI
			01	E1	04AF	BBC	#IDF_V_REMOTE,- ;SKIP IF NOT REMOTE ACCESS
		11	5E	A8	04B1		IDF_Q_FLAG(R8),36\$;
				04B4	724		
				04B9	725	CLRB	FABSV_NAM,FABSL_FOP(AP) ;CLEAR OPEN BY NAM BLOCK FLAG
50	68	A8	DO	04B9	726	MOVL	IDF_L_FILENAME(R8),R0 ;GET ADDRESS OF ASCII FILENAME
34	AC	80	90	04BD	727	MOVB	(R0)+,FABSB_FNS(AP) ;GET LENGTH OF FILE NAME
2C	AC	50	DO	04C1	728	MOVL	R0,FABSL_FNA(AP) ;GET ADDRESS OF FILE NAME
				04C5	729	SOPEN	FAB=(AP) ;OPEN PREVIOUS INPUT
	05	50	E8	04CE	730	BLBS	R0,38\$;BRANCH IF SUCCESSFUL
	FB2C		30	04D1	731	BSBW	DCL\$ERRORMSG ;REPORT ERROR MESSAGE
		3D	11	04D4	732	BRB	40\$;
				04D6	733	CLRB	FABSV_NAM,FABSL_FOP(AP) ;REMOVE OPEN BY NAME BLOCK FLAG
04	A8	02	AC	B0	04DB	MOVW	FABSW_IFI(AP),IDF_W_INPIFI(R8) ;SET NEW INPUT IFI
		02	A6	B4	04E0	CLRW	RABSW_ISI(R6) ;ZERO PREVIOUS INTERNAL SEQUENCE NUMBER
				04E3	736	\$CONNECT	RAB=TR6) ;CONNECT TO PREVIOUS INPUT
				04EC	737	BLBC	R0,37\$;BRANCH IF UNSUCCESSFUL
1F	18	A6	1C	E1	04EF	BBC	#DEVSV_RND,RABSL_CTX(R6),40\$;SKIP IF NOT A DISK FILE
14	A6	5C	A8	B0	04F4	MOVW	IDF_W_INPRFA+4(R8),RABSW_RFA4(R6) ;COPY RECORD FILE ADDRESS FROM
10	A6	58	A8	DO	04F9	MOVL	IDF_W_INPRFA(R8),RABSW_RFA(R6) ;FROM INDIRECT STACK TO RAB
			13	13	04FE	BEQL	40\$;BR IF PREVIOUS FILE AT TOP OF FILE
	1E	A6	02	90	0500	MOVB	#RABSC_RFA,RABSB_RAC(R6) ;SET ACCESS MODE TO RECORD-FILE ADR
				0504	743	\$FIND	RAB=(R6) ;SET TO NEXT RECORD POSITION
				050D	744	BLBC	R0,37\$;BRANCH IF UNSUCCESSFUL
				0510	745	ASSUME	RABSC_SEQ EQ 0
				0510	746	CLRB	RABSB_RAC(R6) ;SET ACCESS TO SEQUENTIAL
				0513	747		
				0513	748		

```
0513 749 : CLOSE CURRENT OUTPUT FILE IF THE CURRENT OUTPUT FILE IS DIFFERENT
0513 750 : FROM THE PREVIOUS LEVEL.  CREATE SYSS$INPUT AND SYSS$OUTPUT LOGICAL NAMES.
0513 751 :
0513 752 40$:  POPL  R8 ;GET ADDR OF JUST CLOSED IDF FRAME
52 0094 C8 8ED0 0516 753  MOVAB  IDF_W_OUTIF1+IDF_K_LENGTH(R8),R2 ;GET ADDR OF OUTPUT FILE INFO
      FAE2' 30 051B 754  BSBW  DCL$RESTORE_OUTPUT ;RESET OLD SYSS$OUTPUT
58 00BC CB  D0 051E 755  MOVL  PRC_L_IDFLNR(R11),R8 ;GET ADDR OF CURRENT IDF FRAME
      FADA' 30 0523 756  BSBW  DCL$CREATE_IO ;CREATE 'INPUT' AND 'OUTPUT' LOGICAL NAMES
      05 0526 757  RSB
      0527 758
      0527 759 :
      0527 760 : DO NOT OPEN THIS INPUT FILE.  REOPEN THE NEXT ONE.
      0527 761 :
      0527 762 50$:  POPL  R8 ;GET ADDR OF JUST CLOSED IDF FRAME
52 0094 C8 8ED0 0527 762  MOVAB  IDF_W_OUTIF1+IDF_K_LENGTH(R8),R2 ;GET ADDR OF OUTPUT FILE INFO
      FACE' 30 052A 763  BSBW  DCL$RESTORE_OUTPUT ;RESET OLD SYSS$OUTPUT
58 00BC CB  D0 0532 764  MOVL  PRC_L_IDFLNR(R11),R8 ;GET ADDR OF CURRENT IDF FRAME
      58 DD 0537 765  PUSHL  R8 ;SAVE THAT ADDRESS
      FE70 31 0539 766  BRW  10$ ;REOPEN NEXT INPUT FILE
      767
```



```
053C 769 .SBTTL SAVE VERIFICATION STATE
053C 770
053C 771 :+ SAV_EXE_ONLY - SAVE EXECUTE ONLY VERIFICATION STATE
053C 772
053C 773 : THIS ROUTINE CHECKS IF PROCEDURE THAT IS ABOUT TO BE EXECUTED IS THE FIRST
053C 774 : EXECUTE-ONLY PROCEDURE ENCOUNTERED SO FAR. IF SO, IT SAVES THE VERIFICATION
053C 775 : STATES AND THE LEVEL NUMBER.
053C 776
053C 777 : INPUTS:
053C 778
053C 779 : R11 - ADDRESS OF PROCESS WORK AREA
053C 780
053C 781 : OUTPUTS:
053C 782
053C 783 : NONE
053C 784 :-
053C 785
053C 786 SAV_EXE_ONLY:
053C 787 MOVL R6,-(SP) ;SAVE WORK REGISTER
2B 16 AC 01 E0 053F 788 BBS #FAB$V GET,FAB$B FAC(AP),30$ ;SKIP IF READ ACCESS
012D CB 95 0544 789 TSTB PRC_B_EXONLYL(R11) ;FIRST ONE ENCOUNTERED?
25 12 0548 790 BNEQ 30$ ;NO, JUST SKIP IT
054A 791
054A 792 BICB #PRC V SAVCMDV!PRC V_SAVINGV,- ;PRESET SAVED VERIF. FLAGS
012C CB 03 8A 054A 792 PRC_B_OUTFLAGS(R11)
56 D4 054C 793
05 68 AB 07 E1 054F 794 CLRL R6 ;TURN OFF IMG. VERIF.
0551 795 BBC #PRC V VERIFY,PRC_W_FLAGS(R11),10$ ;SKIP IF NO VERIFY
0556 796 SETBIT PRC V SAVCMDV,PRC_B_OUTFLAGS(R11) ;SET CMD. VERIFY
05 00AF CB 07 E1 055B 797 10$: BBC #PRC V VERIMAGE,PRC_B_FLAGS2(R11),20$ ;SKIP IF NO VERIFY
0561 798 SETBIT PRC V SAVINGV,PRC_B_OUTFLAGS(R11) ;SET IMG. VERIFY
012D CB 5C AB 90 0566 799 20$: BSBW DCL$SETVERIFY_IMAGE
56 8E D0 056F 800 MOVB PRC_L_INDEPTH(R11),PRC_B_EXONLYL(R11) ;SAVE LEVEL NUMBER
05 0572 801 30$: MOVL (SPT)+,R6 ;RESTORE WORK REGISTERS
0579 802 STATUS NORMAL ;ALWAYS EXIT WITH SUCCESS
803 RSB ;EXIT
```

```
057A 805 .SBTTL RESTORE VERIFICATION STATE
057A 806
057A 807
057A 808 RES_EXE_ONLY - RESTORE EXECUTE ONLY VERIFICATION STATE
057A 809
057A 810 THIS ROUTINE CHECKS IF THE FRAME CURRENTLY BEING UNSTACKED IS THE FIRST
057A 811 EXE-ONLY PROCEDURE ENCOUNTERED. IF SO, IT RESTORES THE VERIFICATION STATES
057A 812 TO WHAT THEY WERE PRIOR TO THE EXECUTE ONLY PROCEDURE.
057A 813
057A 814 INPUTS:
057A 815
057A 816 R11 = ADDRESS OF PROCESS WORK AREA
057A 817
057A 818 OUTPUTS:
057A 819
057A 820 NONE
057A 821
057A 822
057A 823 RES_EXE_ONLY:
057A 824 MOVL R6,-(SP) ;SAVE WORK REGISTER
057A 825 CMPB PRC_L_INDEPTH(R11),- ;IS THIS 1ST EX-ONLY LEVEL?
0580 826 PRC_B_EXONLYL(R11)
0583 827 BNEQ 30$ ;NO, SKIP THIS ONE
0585 828
0585 829 CLRB PRC_B_EXONLYL(R11) ;CLEAR EXE-ONLY FLAG
0589 830 CLRB PRC_V_VERIFY,PRC_W_FLAGS(R11) ;INIT. CMD. VERIF. FLAG
058E 831 BBC #PRC_V_SAVCMDV,PRC_B_OUTFLAGS(R11),10$ ;SKIP IF NO VERIFY
0594 832 SETBIT PRC_V_VERIFY,PRC_W_FLAGS(R11) ;SET CMD. VERIFICATION
0599 833
0599 834 10$: CLRL R6 ;ASSUME NO IMAGE VERIFICATION
059B 835 BBC #PRC_V_SAVINGV,PRC_B_OUTFLAGS(R11),20$ ;SKIP IF NO VERIFY
05A1 836 INCL R6 ;SET FLAG TO SET IMG. VERIFICATION
05A3 837 20$: CLRB PRC_V_VERIFY,PRC_B_FLAGS2(R11) ;SYNC FLAG WITH LAST SET STATE
05A9 838 BSBW DCL$SETVERIFY_IMAGE ;SET/RESET IMAGE VERIFICATION
05AC 839 30$: MOVL (SP)+,R6 ;RESTORE WORK REGISTER
05AF 840 STATUS NORMAL ;ALWAYS SIGNAL SUCCESS
05B6 841 RSB ;EXIT
05B7 842
05B7 843 .END
```

7E 56 D0
5C AB 91
012D CB
27 12
012D CB 94
05 012C CB 02 E1
02 012C CB 56 D4
03 E1
56 D6
FA54' 30
56 8E D0
05

INDIRECT
Symbol table

- INDIRECT FILE MANIPULATION ROUTINES

15-SEP-1984 23:55:59 VAX/VMS Macro V04-00
4-SEP-1984 23:41:10 [DCL.SRC]INDIRECT.MAR;1Page 20
(8)

```

SS.TMP1      = 00000001
SS.TMP2      = 00000066
$ST1         = 00000000
CLIS_DEFOVF  = 00038028
CLIS_IVQUAL  = 00038240
CLIS_IVVALU  = 00038088
CLIS_NORMAI  = 00030001
CLIS_STKO    = 00038128
CLIS_SYMOV   = 00038138
CLIS_USGOTO  = 00038148
DCL$ALLDYNMEM ***** X 02
DCL$ALLOCSYM ***** X 02
DCL$COMPRESS ***** X 02
DCL$CREATE_IO ***** X 02
DCL$DEADYNMEM ***** X 02
DCL$DEALGOTO ***** X 02
DCL$DEALLOCSYM ***** X 02
DCL$DEFINE_P1_TO_P8 00000105 RG 02
DCL$DISABLE ***** X 02
DCL$ERRORMSG ***** X 02
DCL$GETDVAL ***** X 02
DCL$MARK ***** X 02
DCL$MARKEDTOKEN ***** X 02
DCL$MOVCHAR ***** X 02
DCL$MOVTKN ***** X 02
DCL$ONCTLYRST ***** X 02
DCL$ONRESET ***** X 02
DCL$OPEN_CREATE ***** X 02
DCL$OPEN_OUTPUT ***** X 02
DCL$PROCFILE ***** X 02
DCL$PUSHPROC 00000139 RG 02
DCL$RESTORE_OUTPUT ***** X 02
DCL$RUNDOWN ***** X 02
DCL$SETCHAR ***** X 02
DCL$SETNBK ***** X 02
DCL$SETVERIFY_IMAGE ***** X 02
DCL$SET_STATUS ***** X 02
DCL$STACKIND 00000027 RG 02
DCL$T_DEFONTXT ***** X 02
DCL$UNSTACK 00000319 RG 02
DEVSV_RND    = 0000001C
FABS$B_DNS   = 00000035
FABS$B_FAC   = 00000016
FABS$B_FNS   = 00000034
FABS$B_NAT   = 0000001E
FABS$B_RFM   = 0000001F
FABS$B_SHR   = 00000017
FABS$C_VFC   = 00000003
FABS$L_DEV   = 00000040
FABS$L_DNA   = 00000030
FABS$L_FNA   = 0000002C
FABS$L_FOP   = 00000004
FABS$L_NAM   = 00000028
FABS$M_EXE   = 00000080
FABS$M_GET   = 00000002
FABS$M_INP   = 00080000
FABS$M_NAM   = 01000000

```

```

FABS$M_PPF   = 00040000
FABS$M_PRN   = 00000004
FABS$V_GET   = 00000001
FABS$V_NAM   = 00000018
FABS$W_IFI   = 00000002
IDF_B_OUTFLAGS 00000038
IDF_C_LENGTH 00000074
IDF_K_LENGTH 00000074
IDF_L_FILENAME 00000068
IDF_L_INPRABCTX 0000000C
IDF_L_LNK 00000000
IDF_L_ONCTLY 00000060
IDF_L_ONERROR 00000008
IDF_L_OUTRABCTX 00000024
IDF_L_SEARCHCTX 00000064
IDF_Q_LABEL 00000018
IDF_Q_LOCAL 00000010
IDF_T_INPDVI 0000003C
IDF_T_OUTDVI 00000028
IDF_V_INPCCL = 00000001
IDF_V_INPDPM = 00000000
IDF_V_REMOTE = 00000001
IDF_W_FLAG 0000005E
IDF_W_INPDID 00000052
IDF_W_INPFID 0000004C
IDF_W_INPIFI 00000004
IDF_W_INPRFA 00000058
IDF_W_ONLEVEL 00000006
IDF_W_OUTIFI 00000020
IDF_W_OUTISI 00000022
INPFICE 00000000 R 02
LNMS$PROCESS 00000014 R 02
LOG$C_PROCESS = 00000002
NAMS$B_DEV = 00000039
NAMS$B_DIR = 0000003A
NAMS$B_ESL = 0000000B
NAMS$B_ESS = 0000000A
NAMS$B_NAME = 0000003B
NAMS$B_NODE = 00000038
NAMS$B_NOP = 00000008
NAMS$B_RSL = 00000003
NAMS$B_RSS = 00000002
NAMS$B_TYPE = 0000003C
NAMS$B_VER = 0000003D
NAMS$C_MAXR = 000000FF
NAMS$L_ESA = 0000000C
NAMS$L_FNB = 00000034
NAMS$L_RSA = 00000004
NAMS$M_PWD = 00000001
NAM$T_DVI = 00000014
NAMS$V_CNCL_DEV = 0000000C
NAMS$V_NODE = 00000011
NAMS$V_PWD = 00000000
NLAO 00000020 R 02
OUTQUAL 00000004 R 02
PRC_B_CONTINUE 000000F3
PRC_B_DEFRADIX 000000AE

```


INDIRECT
Symbol table

F 16
- INDIRECT FILE MANIPULATION ROUTINES

15-SEP-1984 23:55:59 VAX/VMS Macro V04-00
4-SEP-1984 23:41:10 [DCL.SRC]INDIRECT.MAR;1

Page 21
(8)

```

PRC_B_EXMDEPMOD      000000AD
PRC_B_EXMDEPWID      000000AC
PRC_B_EXONLYL        0000012D
PRC_B_FLAGS2         000000AF
PRC_B_IMGFLAG        00000078
PRC_B_OUTFLAGS       0000012C
PRC_B_PROMPTLEN      000000F0
PRC_C_LENGTH         00000534
PRC_G_COMMANDS       00000133
PRC_G_PROMPT         000000F4
PRC_K_LENGTH         00000534
PRC_L_CURRKEY        00000048
PRC_L_EXMDEPADR      000000A8
PRC_L_EXTARG         00000094
PRC_L_EXTBLK         0000008C
PRC_L_EXTCOD         0000009C
PRC_L_EXTHND         00000090
PRC_L_EXTPRM         00000098
PRC_L_IDFLNK         0000008C
PRC_L_IMGACTSTS      00000080
PRC_L_INDCLOCK       0000007C
PRC_L_INDEPTH        0000005C
PRC_L_INDFAB         0000001C
PRC_L_INDINPRAB      00000014
PRC_L_INOUTRAB       00000018
PRC_L_INPRAB         00000008
PRC_L_LASTKEY        0000004C
PRC_L_LSTSTATUS      00000080
PRC_L_ONCTLY         00000088
PRC_L_ONERROR        0000006C
PRC_L_OUTOFBAND      000000B4
PRC_L_OUTRAB         00C0000C
PRC_L_OUTRABCTX      00000118
PRC_L_PPFLIST        00000070
PRC_L_RECALLPTR      0000012F
PRC_L_RESTART        00000058
PRC_L_SAVAP          00000000
PRC_L_SAVFP          00000004
PRC_L_SEVERITY       00000050
PRC_L_SPWN           000000C0
PRC_L_STACKLM        000000A4
PRC_L_STACKPT        000000A0
PRC_L_STATUS         00000054
PRC_L_STS            00000084
PRC_L_STV            00000088
PRC_L_SYMBOL         00000060
PRC_L_TMBX           00000074
PRC_L_TRMLIST        00000010
PRC_Q_ALLOCREG       00000020
PRC_Q_COMMAND        000000E0
PRC_Q_FLUSHTIME      000000D0
PRC_Q_GLOBAL         00000028
PRC_Q_IMAGE_NAME     000000D8
PRC_Q_KEYPAD         00000040
PRC_Q_LABEL          00000030
PRC_Q_LOCAL          00000038
PRC_Q_SAVEPRIV       000000E8

```

```

PRC_T_OUTDVI         0000011C
PRC_V_GOTO           = 00000004
PRC_V_SAVCMDV        = 00000002
PRC_V_SAVIMGV        = 00000003
PRC_V_VERIFY         = 00000007
PRC_V_VERIMAGE       = 00000007
PRC_W_ASTIOSB        000000C6
PRC_W_ASTRETN        000000C8
PRC_W_ASTSTATUS      000000C4
PRC_W_ATTMBX         0000007A
PRC_W_FLAGS          00000068
PRC_W_INPCAN         00000064
PRC_W_ONLEVEL        0000006A
PRC_W_OUTIFI         00000114
PRC_W_OUTISI         00000116
PRC_W_OUTMBXCHN      000000CA
PRC_W_OUTMBXREF      000000CE
PRC_W_OUTMBXSIZ      000000CC
PRC_W_PMPTCTRL       000000F1
PRC_W_WAITIOSB       00000066
PRD_C_LENGTH         00000214
PRD_C_XLENGTH        00000244
PRD_G_ALTINPRAB      000000F4
PRD_G_ALTOURAB       00000138
PRD_G_FAB            00000000
PRD_G_INPRAB         000000B0
PRD_G_NAM            00000050
PRD_G_OUTRAB         0000017C
PRD_G_TRMLIST        000001E4
PRD_G_XABTRM         000001C0
PRD_K_LENGTH         00000214
PRD_K_XLENGTH        00000244
PRD_T_OUTDVI         00000214
PRD_T_OUTFNM         00000230
PRD_W_OUTDID         0000022A
PRD_W_OUTFID         00000224
PSL$C_SUPER          = 00000002
PSL$C_USER           = 00000003
PTR_B_LEVEL          00000004
PTR_B_NUMBER         00000005
PTR_B_PARMCNT        00000006
PTR_B_VALUE          00000000
PTR_C_LENGTH         0000000C
PTR_K_LENGTH         0000000C
PTR_K_PARAMETER      = 00000003
PTR_L_DESCR          00000000
PTR_L_ENTITY         00000008
RAB$B_RAC            = 0000001E
RAB$C_RFA            = 00000002
RAB$C_SEQ            = 00000000
RAB$C_CTX            = 00000018
RAB$C_FAB            = 0000003C
RAB$V_PPF_IND        = 0000000E
RAB$W_ISI            = 00000002
RAB$W_RFA            = 00000010
RAB$W_RFA4           = 00000014
RES_EXE_ONLY         0000057A R

```

INDIRECT
Symbol table

G 16
- INDIRECT FILE MANIPULATION ROUTINES

15-SEP-1984 23:55:59 VAX/VMS Macro V04-00
4-SEP-1984 23:41:10 [DCL.SRC]INDIRECT.MAR;1

Page 22
(8)

RMSS_EOF
SAV_EXE_ONLY
SETIND
SS\$NORMAL
STKXIT
SYMBOLS
SYM_B_FLAGS
SYM_B_NONUNIQUE
SYM_B_TYPE
SYM_K_STRING
SYM_L_BL
SYM_L_FL
SYM_T_SYMBOL
SYM_W_SIZE
SYSSCLOSE
SYSSCONNECT
SYSSDELLNM
SYSSDELLOG
SYSSFIND
SYSSOPEN
SYSSPARSE
SYS_INPUT_NAME
UNSTACK
WRK_B_CMDOPT
WRK_B_MAXPARM
WRK_B_MINPARM
WRK_B_PARMCNT
WRK_B_PARMSUM
WRK_B_RECALLCNT
WRK_B_VALLEV
WRK_B_VERBTYP
WRK_C_LENGTH
WRK_G_BUFFER
WRK_G_INPBUF
WRK_G_RESULT
WRK_K_LENGTH
WRK_L_CHARPTR
WRK_L_DISALLOW
WRK_L_ERRORRTN
WRK_L_EXPANDPTR
WRK_L_IMAGE
WRK_L_MARKPTR
WRK_L_PAROUT
WRK_L_PMPTADDR
WRK_L_PROMPTRTN
WRK_L_PROPTR
WRK_L_QUABLK
WRK_L_READRTN
WRK_L_RECALLPTR
WRK_L_RSLEND
WRK_L_RSLNXT
WRK_L_SAVAP
WRK_L_SAVFP
WRK_L_SAVSP
WRK_L_SIGNALRTN
WRK_L_SPECRTN
WRK_L_TAB_VEC

***** X 02
0000053C R 02
00000370 R 02
***** X 02
0000034F R 02
= 00000008
0000000B
0000000B
0000000A
= 00000000
00000004
00000000
0000000C
00000008
***** GX 02
***** GX 02
***** GX 02
***** GX 02
***** GX 02
***** GX 02
***** GX 02
0000000A R 02
00000393 R 02
FFFFFFFFC3
FFFFFFFFD0
FFFFFFFFD1
FFFFFFFFCE
FFFFFFFFCF
FFFFFFFFC5
FFFFFFFFC4
FFFFFFFFC2
FFFFFF486
FFFFFF492
FFFFFF896
FFFFFF9B6
FFFFFF486
FFFFFF48E
FFFFFFFE6
FFFFFF9AE
FFFFFF486
FFFFFFFE2
FFFFFF48A
FFFFFFFD2
FFFFFF9A2
FFFFFF9A6
FFFFFFFC6
FFFFFFFCA
FFFFFF9AA
FFFFFFFEA
FFFFFFFB6
FFFFFFFBA
FFFFFFF8
FFFFFFFC
FFFFFFF4
FFFFFFFD6
FFFFFF9B2
FFFFFFFDE

WRK_L_VERB
WRK_M_COMMAND
WRK_V_COMMAND
WRK_V_QUOTE
WRK_W_FLAGS
WRK_W_FLAGS2
WRK_W_IMGCHAN
WRK_W_PMPTLEN
SS

FFFFFFBE
= 00000002
= 00000001
= 00000004
FFFFFFFF0
FFFFFFFF2
FFFFFFFFE
= 000000EF

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	FFFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DCL\$ZCODE	000005B7 (1463.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.08	00:00:02.40
Command processing	83	00:00:00.67	00:00:05.41
Pass 1	355	00:00:15.13	00:00:44.00
Symbol table sort	0	00:00:01.61	00:00:03.90
Pass 2	157	00:00:03.15	00:00:10.87
Symbol table output	34	00:00:00.27	00:00:01.15
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	640	00:00:20.93	00:01:07.75

The working set limit was 1500 pages.

76452 bytes (150 pages) of virtual memory were used to buffer the intermediate code.

There were 60 pages of symbol table space allocated to hold 1153 non-local and 47 local symbols.

843 source lines were read in Pass 1, producing 18 object records in Pass 2.

62 pages of virtual memory were used to define 43 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[SYSLIB]SYSBLDMLB.MLB;1	0
_\$255\$DUA28:[DCL.OBJ]DCL.MLB;1	14
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	1
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	19
TOTALS (all libraries)	34

1417 GETS were required to define 34 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:INDIRECT/OBJ=OBJ\$:INDIRECT MSRC\$:INDIRECT/UPDATE=(ENH\$:INDIRECT)+EXECMLS/LIB+LIB\$:DCL/LIB+SYSSLIBRARY:SYSBLDMLB/LIB

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY